Amendments to the Specification

Please replace the paragraph on Page 1, lines 5 - 10 with the following marked-up replacement paragraph:

-- The present invention is related to U. S. Patent _____, titled "Array-Based Extensible Document Storage Format" (serial number ~~09/_____), referred~~ 09/652,296), referred to herein as the "first related invention", and U. S. Patent _____, titled "High-Performance Extensible Document Transformation" (serial number ~~09/_____), filed~~ 09/653,080), filed concurrently herewith. These related inventions are commonly assigned to International Business Machines Corporation (IBM), and are hereby incorporated herein by reference. --

Please replace the paragraph on Page 2, lines 8 - 18 with the following marked-up replacement paragraph:

-- Business and consumer use of distributed computing, also commonly referred to as network computing, has gained tremendous popularity in recent years. In this computing model, the data and/or programs to be used to perform a particular computing task typically reside on (i.e. are "distributed" among) more than one computer, where these multiple computers are connected by a network of some type. The Internet, and the part of the Internet known as the World Wide ~~Web (hereinafter, "Web", are~~ Web (hereinafter, "Web"), are well-known examples of this type of environment wherein the multiple computers are connected using a public network. Other types of network environments in which distributed computing may be used include intranets, which are typically private networks accessible to a restricted set of users (such as

employees of a corporation), and extranets (e.g., a corporate network which is accessible to other users than just the employees of the company which owns and/or manages the network, such as the company's business partners). --

Please replace the paragraph that begins on Page 3, line 13 and carries over to Page 4, line 5 with the following marked-up replacement paragraph:

-- The syntax of XML is extensible and flexible, and allows document developers to create tags to convey an explicit nested tree document structure (where the structure is determined from the relationship among the tags in a particular document). Furthermore, document developers can define their own tags which may have application-specific semantics. Because of this extensibility, XML documents may be used to specify many different types of information, for use in a virtually unlimited number of contexts. It is this extensibility and flexibility which is, in large part, responsible for the popularity of XML. (A number of XML derivative notations have been defined, and continue to be defined, for particular purposes. "VoiceXML" is an example of one such derivative. References herein to "XML" are intended to include XML derivatives and semantically similar notations such as derivatives of the Standard Generalized Markup Language, or "SGML", from which XML was derived. Refer to ISO 8879, "Standard Generalized Markup Language (SGML)", (1986) for more information on SGML. Refer to "Extensible Markup Language (XML), W3C Recommendation 10-February-1998" which is available from the World Wide Web Consortium, or "W3C", for ~~on the World Wide Web at http://www.w3.org/TR/1998/REC-xml-19980210, for~~ more information on XML.) --

Please replac the paragraph on Page 18, lines 2 - 13 with the following marked-up replacement paragraph:

-- Fig. 4A illustrates a simple structured document 400 which is represented in the existing XML notation. This document contains 6 elements which are organized in a 3-level hierarchy. The node having element name "root_element" 402 is the root node, being at the highest level of the hierarchy. This node has 2 child nodes, having element names "level_one_element1" 410 and "level_one_element2" 420. Node "level_one_element1" 410 also has 2 child nodes, which are the nodes having element names "level_two_element11" 412 and "level_two_element12" 414, and ~~node "level_two_element2" 420~~ node "level_one_element2" 420 has a single child node having element name "level_two_element21" 422. A tree structure 430 representing document 400 is shown in Fig. 4B, where the tags for the 6 elements are depicted inside rectangular shapes representing nodes of the tree and the data content corresponding to each node is shown inside an ellipse. This interpretation of an XML document 400 and its corresponding tree structure 430 are well known in the art. --

Please replace the paragraph on Page 24, lines 1 - 10 with the following marked-up replacement paragraph:

-- The information for each attribute in the attribute list is also preferably delimited using a comma. Within each attribute's information, a period is preferably used as a delimiter. Referring to the example in Fig. 4A, node [[B1]] 410 has 2 attributes. The first has the attribute name "id" and the attribute value "1". Thus, the length of the attribute name is 2, and the length of the attribute value is 1. Again using zero-based counting, the first attribute represented in the

attribute list for the node shown at 410 is therefore specified as "0.2.2.1", meaning that the name of the attribute is found in the data buffer starting at position 0 for a length of 2 characters, and the value is found starting at position 2 for a length of 1. As shown in Fig. 4C (see reference number 480), the data buffer is preferably stored at the end of the mXML document. A parser can therefore avoid scanning these characters during the parsing process when they are not needed. –

Please replace the paragraph on Page 24, lines 11 - 17 with the following marked-up replacement paragraph:

-- The second attribute of node 410 ~~of B1's attributes~~ in this example has the name "name" and the value "1". The information for this second attribute is therefore specified using the syntax "3.4.7.1", meaning that the attribute's name is found in the data buffer starting at position 3 for a length of 4 characters and its value is found starting at position 7 for a length of 1. If a node has more than 2 attributes, this dot-delimited syntax is used for each such attribute, and is separated from the other attribute specifications for this node using commas as delimiters. As with the child list syntax, if a node has no attributes, the absence is indicated by specifying an empty attribute list. –

Please replace the paragraph that begins on Page 24, line 19 and carries over to Page 25, line 5 with the following marked-up replacement paragraph:

-- While the syntax used in the preferred embodiment refers to the data buffer using starting positions and length values, as described for the attribute names and values of the node at

412, in an alternative syntax the starting and ending positions within the data buffer may be used. Thus, the specification for the first attribute of the node at 412 would be expressed as "0.1.2.2", meaning that the attribute name begins at position 0 and ends at position 1, and the attribute value begins and ends at position 2. Similarly, the specification [[fro]] for the second attribute would be expressed as "3.6.7.7". Use of length values, as selected for the syntax of the preferred embodiment, will in general require slightly less space than use of ending positions. --

Please replace the paragraph on Page 25, lines 4 - 14 with the following marked-up replacement paragraph:

-- Thus, it can be seen that the structure of an mXML document is explicitly specified within the document. This information can be used to build a Document Object Model ("DOM") tree, if desired. The DOM tree can then be processed as in the prior art. Alternatively, the mXML document notation can be traversed directly, for example to locate information about a particular node, to determine the overall structure of the document, or to otherwise operate upon the mXML document. The mXML document may be stored using the array-based extensible document storage format described in the first related invention, resulting in further processing efficiencies (as described therein) when operating on a document. (DOM is published as a Recommendation of the World Wide Web Consortium ("W3C"), titled "Document Object Model (DOM) Level 1 Specification, Version 1.0" (1998), which is available from the W3C.) ~~and available on the Web at http://www.w3.org/TR/REC-DOM-Level-1. "DOM" is a trademark of Massachusetts Institute of Technology.)~~ --

Please replace the paragraph that begins on Page 27, line 6 and carries over to Page 28, line 12 with the following marked-up replacement paragraph:

-- The XML notation includes a number of notational elements which are not strictly necessary for data-centered document specification. An XML subset referred to as "SML", for "Simple Markup Language", is currently under discussion in the technical community. This XML subset proposes use of a core set of XML syntax, and omission of features including attributes, processing instructions, etc. See, for example, a Web-published article entitled "SML: Simplifying XML", which [[is]] was written by Robert E. La Quey and published November 24, 1999. The ~~is located at http://www.xml.com/pub/1999/11/sml/index.html (published 11/24/99).~~ ~~The~~ preferred mXML syntax which is described herein provides support for one core set of XML notational elements (although not identical to the core set proposed for SML), where the basic node types include elements and attributes. More complicated XML documents containing additional node types can be supported by extending this preferred mXML syntax, where those additional node types include comments, processing instructions, CDATA, entity, entity reference, and document type nodes. In a preferred technique for specifying this extended mXML syntax, "text" nodes are added to an mXML document to refer to the actual node content. A node specification for a node type such as those just listed preferably occurs in-line within the mXML document, in the same relative location where it appears in a corresponding XML document. This node specification preferably comprises a null value in place of the node name; a list pointing to one or more child nodes, as is used in the node specifications which have been described, except that the children are now text nodes; an empty attribute list; and a pair of special indicators as the node value specification. The starting position entry within the special indicator

pair is used to denote which type of other node is being represented. For example, a value of -2 may represent a comment, while a value of -3 represents a processing instruction, and so forth. The length entry within the special indicator pair is preferably set to -1. The node specification for each of the child text nodes referenced from the special child list preferably also uses a null name, and a null child list and attribute list. The value entry in this child text node then (1) points to a location within the data buffer where the node's content is stored (preferably as a character string representing all the significant content from the source node), and (2) stores the length of this content. --

Please replace the paragraph on Page 33, lines 7 - 17 with the following marked-up replacement paragraph:

-- Block 670 converts the node's attribute information, if any, and writes this to the mXML buffer, followed by a semi-colon delimiter. For each attribute of the current node that is located in the DOM tree, the attribute's name and value are written to the data buffer in successive locations. The position within the data buffer where the name begins, and its length, are written to the mXML buffer as the first two dot-separated integers of the attribute specification. The data buffer counter that was initialized at Block [[610]] 620 is then incremented by the length of the attribute name. Similarly, the position within the data buffer where the attribute value begins, and its length, are written to the mXML buffer using the dot-separated notation (and after a dot that follows the attribute name's length), and the data buffer counter is incremented by the length of the attribute value. If this node has more than one attribute, a comma is written to the mXML buffer to delimit the dot-separated attribute

specifications. --